Oceanus Documentation

SAFE SPOT

Entertainment Technology Center

Contents

SAFESPOT (KISS CAM)

PRODUCTION OVERVIEW

By using the cameras installed in the EDG, we will engage groups of visitors passing through to participate in an unforgettable experience.

On the EDG ceiling, an arctic sea breeze causes ice and glaciers to form, grabbing viewers' attention.

Visitors will see themselves on the ceiling standing on the ice. If they stand where there is water, their figure falls into the water and disappears, thus encouraging them to stand on the ice.

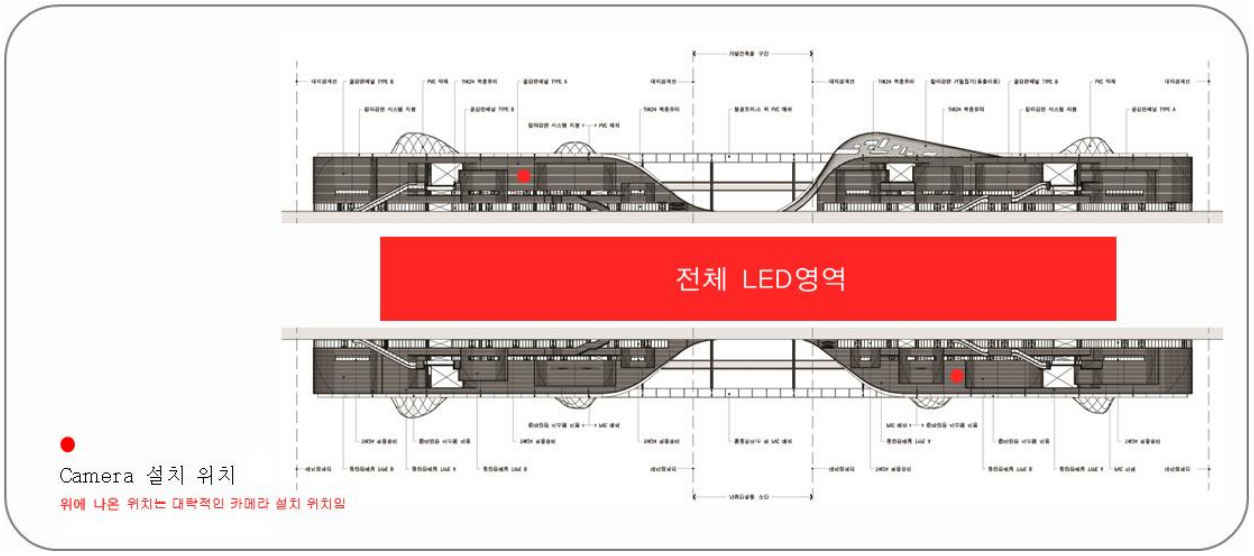
Ever-changing shapes of glacial ice and polar bears interfere with visitors as they attempt to stay where it is safe. The polar bears try to stay on the ice, but their weight breaks the ice, causing the visitors to move to another safe spot and giving the visitors a greater fun factor.

Safe Spot simply gives pleasure to the people directly involved in the activity as well as onlookers in the EDG. Anyone in the audience can have fun.

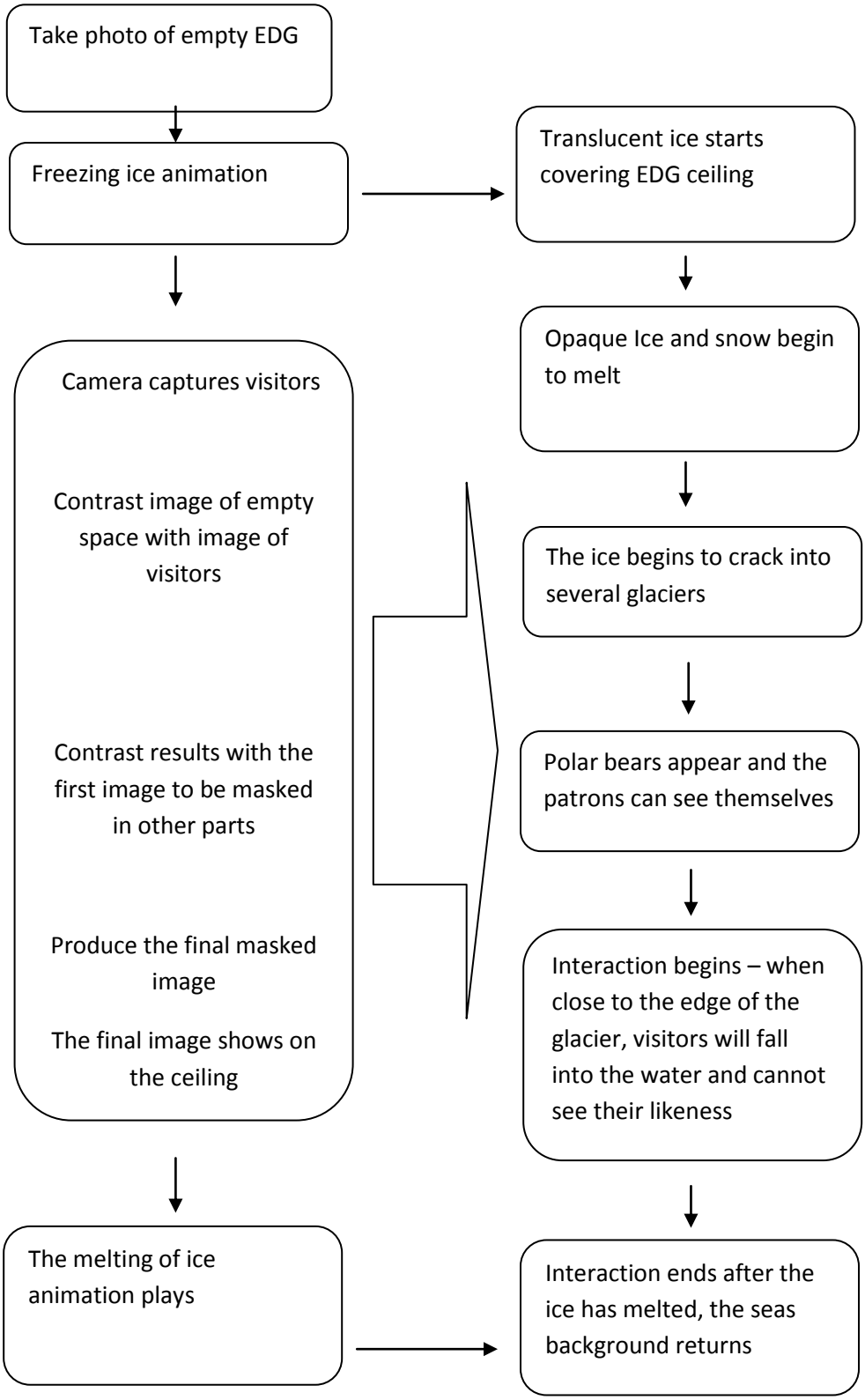
IMPLEMENTATION AREA

The activity will utilize the entire LED ceiling area of the EDG.

Currently, the camera position is inconclusive due to potential changes in the future.



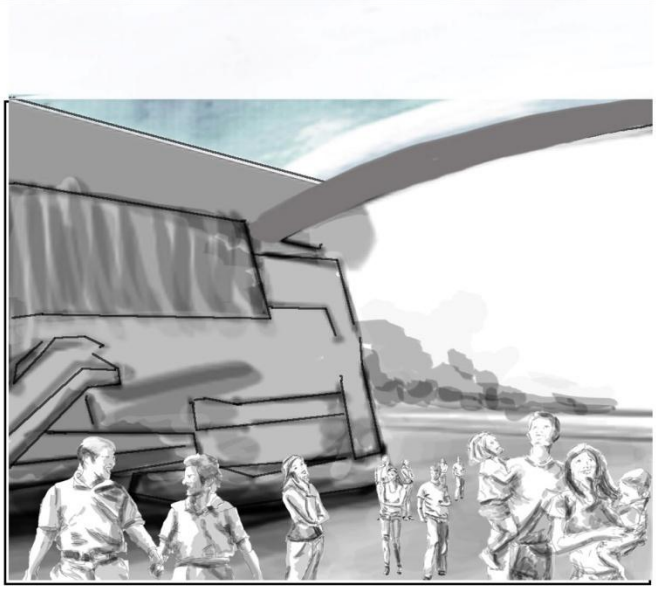
CONFIGURATION



CONFIGURATION DETAILS

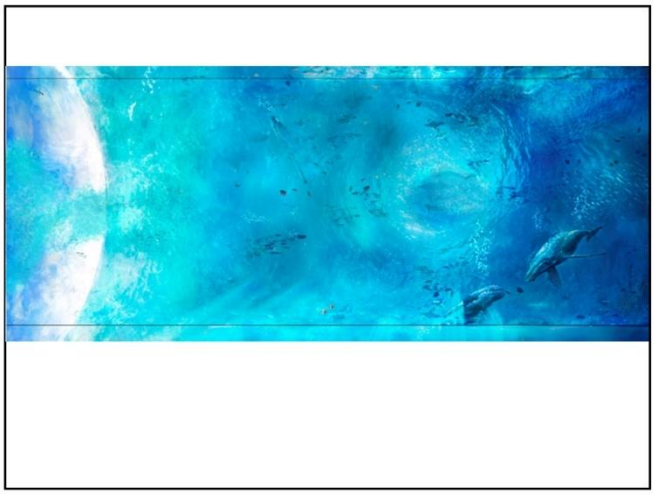
Steps	Sever / Camera	LED
EDG empty space photo	Shoot with the camera, no one in EDG	
Ice Animation	Ice freeze animation play	The sea ice begins to cover on both sides of the EDG ceiling
Ice and snow cover background	Images of empty EDG are contrasted with live footage The differences that occur after the contrast results in a masked image The final image is masked on the screen, revealing visitors Visitors go back to the camera, shooting, and constantly repeating this process	After the translucent ice and snow piles on top, the audience can no longer see the sea background
Cracks in the glacier ice		Ground covered with snow and ice breaks to create several large glaciers
Polar bears appear, visitors can see their appearance		Polar bears emerge and visitors can see their appearance in a live feed
INTERACTION		The ever-changing appearance of ice, glaciers, polar bears, as well as standing access for the audience will cause the audience to move and find safe spots or risk falling into the ocean
Sea background revealed	The melting of ice animation plays	After the sea ice covers the ceiling, it melts from the center out to both ends of the screen

STORYBOARD

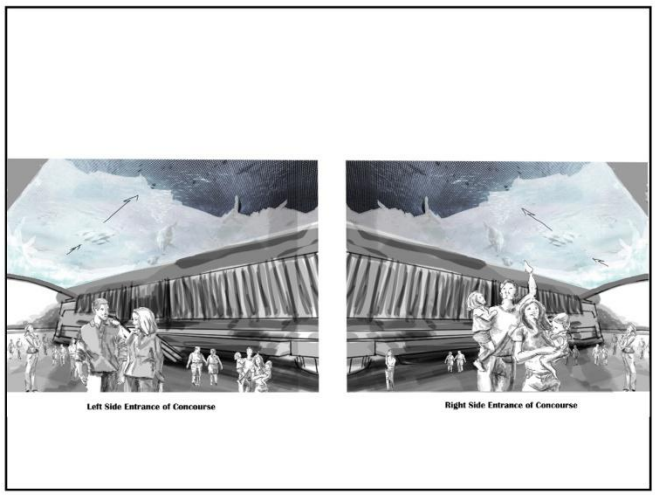


As Families and patrons arrive at the EDG.

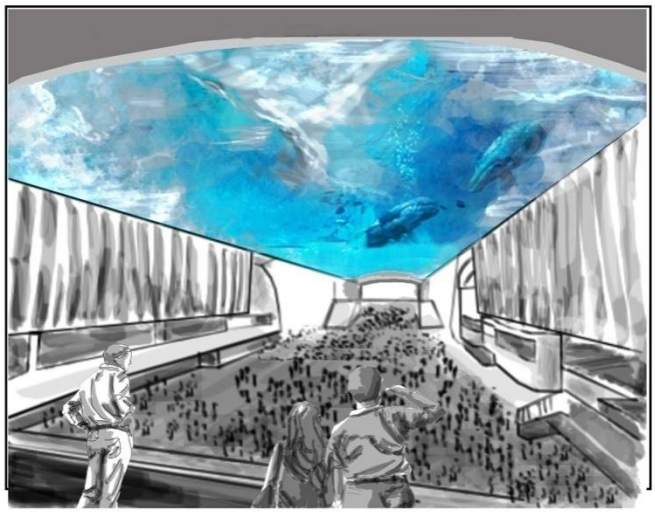
They will notice the ceiling display, which will have the realistic ocean background.

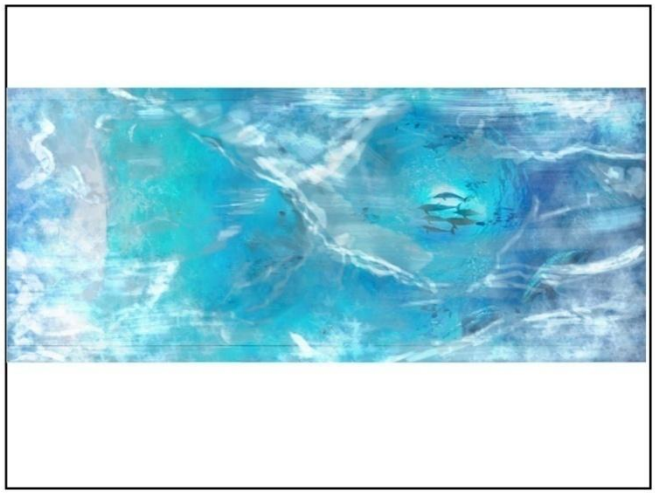


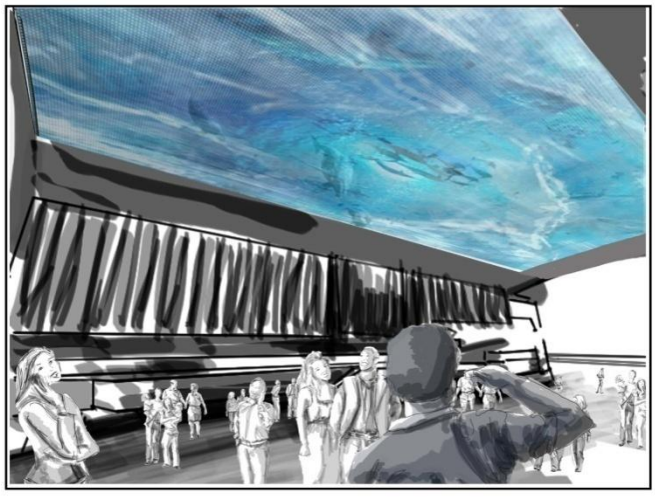
The ocean begins to freeze on the each end of the corridor.



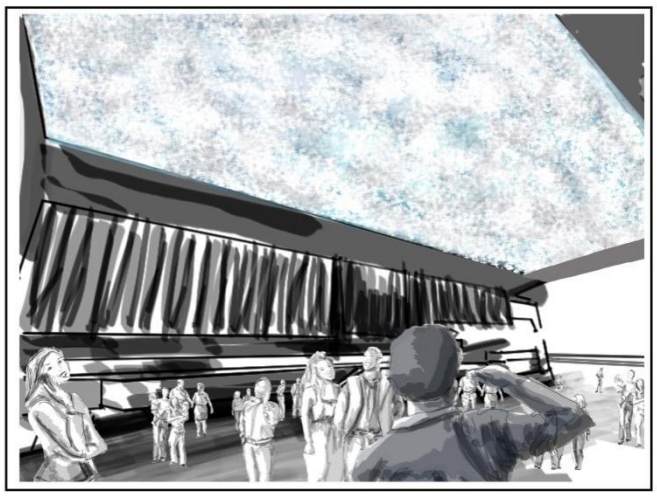
Freezing moves over the entire ocean towards the center of the EDG.



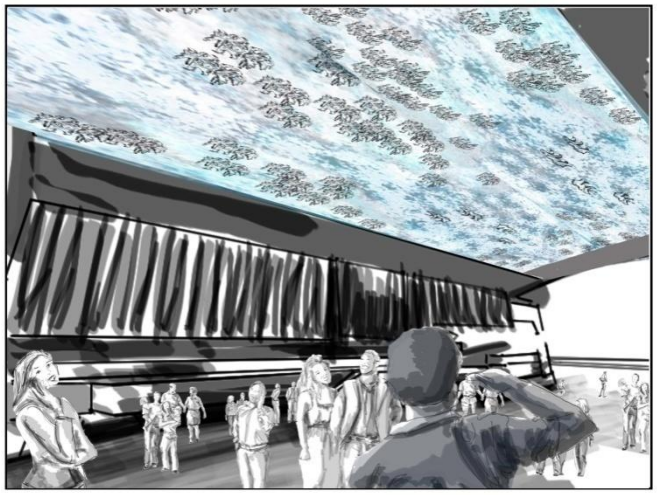




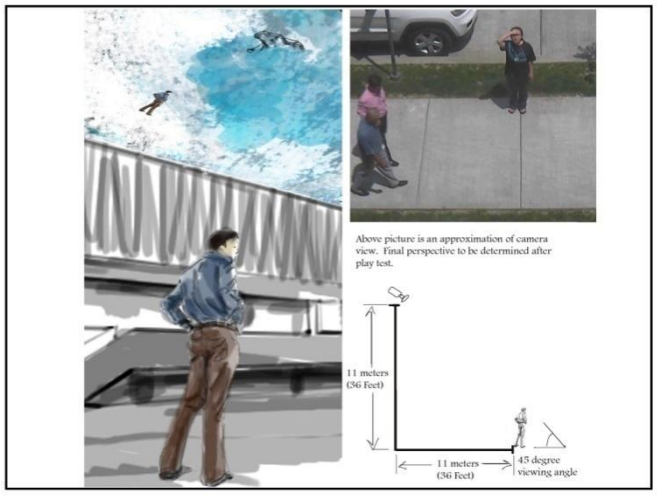
After the entire screen has frozen,



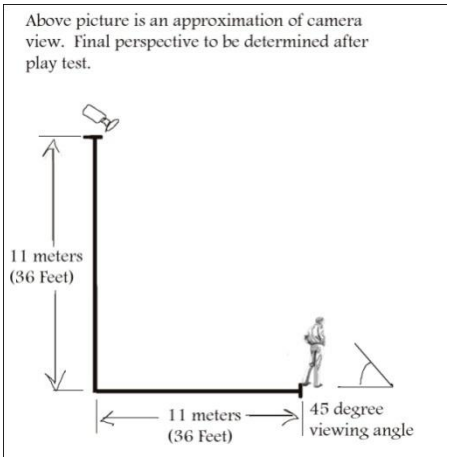
a snow drift covers the ice with snow, causing it to be opaque.

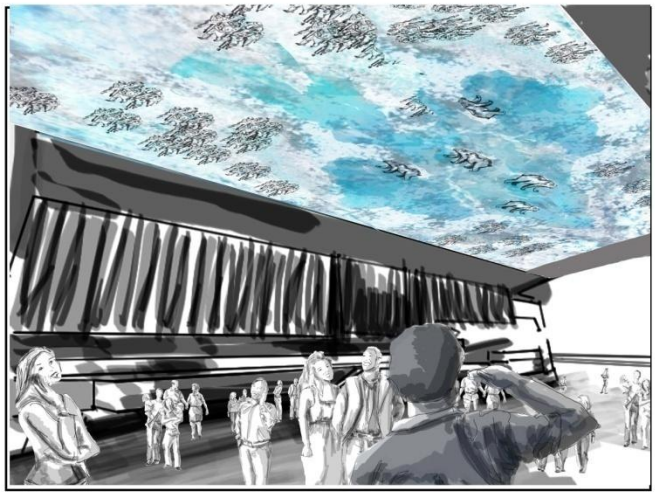


Once the snow has covered the screen, polar bears become visible in the center.



EDG patrons can now see themselves on the snow.





The center of the ice starts to melt and reveal arctic waters and arctic ice...

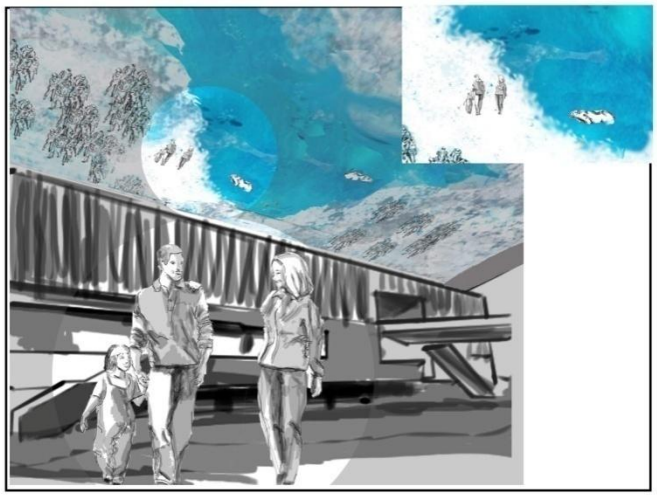
The polar bears fall into the water and swim towards the ice for safety. The interactivity begins.

If a patron steps where there is water, there is a splash and their image disappears. If they step onto the ice, they reemerge.

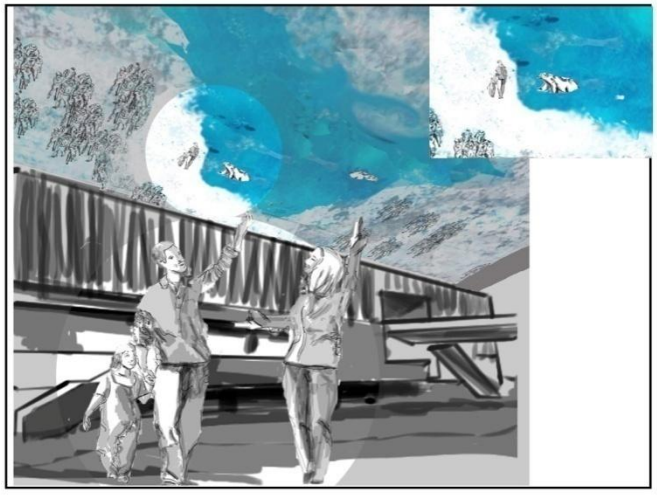
The polar bears try to get onto the ice.

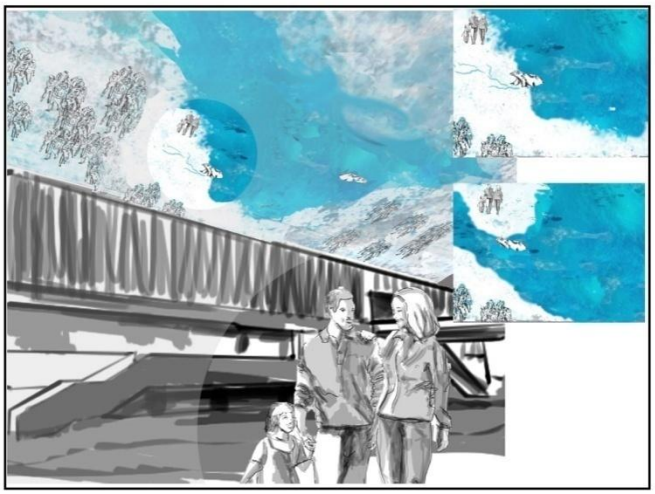
Their weight breaks the melting ice –

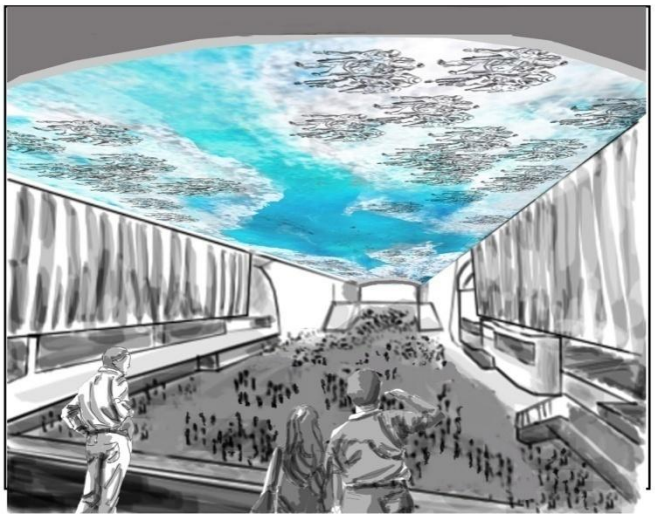
- causing the patrons to fall into the water. Now there is less ice to stand on.



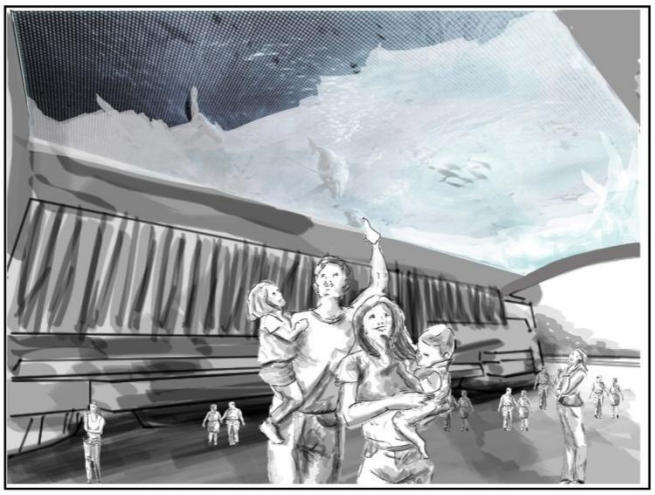
When patrons see themselves onscreen, they will be encouraged to participate.







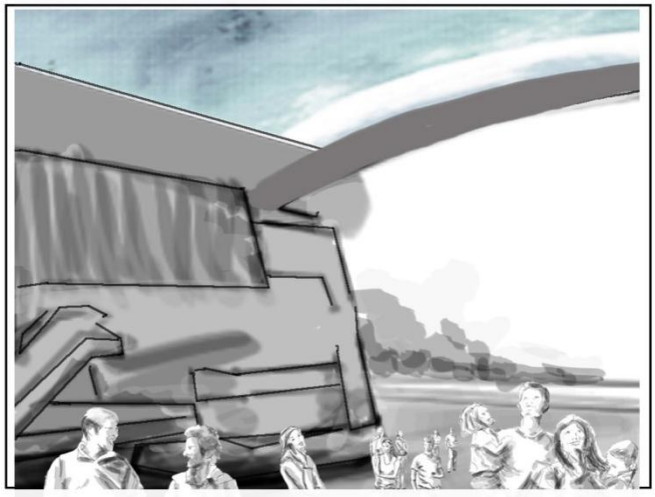
When a polar bear approaches, they will figure out to move to another spot on the ice. Ice patches can break and float away.



At the end of the activity, the snow blows back over the water…

the ice recedes…

and the ocean background returns.



INTRODUCTION

This document provides a comprehensive architectural overview of the Oceanus Safe Spot Unity scene. It outlines the technologies used in implementing the project, and is intended to capture and convey the significant architectural decisions that have been made during the design of the system. It serves as a communication medium between the software architects, developers, deployment specialists, and other team members regarding those decisions.

The Safe Spot Unity scene is being created for the purpose of using a set of cameras to create an interactive experience using virtual imagery. An administrative user will be able to take a photo of an unpopulated area preemptively, run the Unity scene that will call a plugin to perform image differencing against the photo using live camera feeds. The user will only be required to take the unpopulated photo and modify the camera settings to run the application effectively.

PURPOSE

This document will serve two major purposes:

Guide the concrete software implementation of the Oceanus Safe Spot Unity scene.
Assist enterprise architect, operation, platform team, and security auditor conducting an enterprise level best practice and standard validation.

SCOPE

This document describes the architecture of the data structures, communication layers, and configuration of the system. It explains the interactions between the end users of the application, the external and internal interactions between the application and the system, and administration users managing the system.

WHAT THIS DOCUMENT IS NOT

This document does not attempt to describe the existing functionality in detail available within GL's architecture as GL's system is an integration point in the architecture, nor does it attempt to describe the interactions and details of the Twitter subsystem as separate documentation exists for the Twitter component.

ASSUMPTIONS

OpenCV 2.1 and the current version of Unity of the time will be used for development. Future API versions may introduce incompatibilities that will need to be remedied by future updates to the source code.

RELATED DOCUMENTS

OpenCV Documentation
http://opencv.willowgarage.com/wiki/
Unity3D Documentation

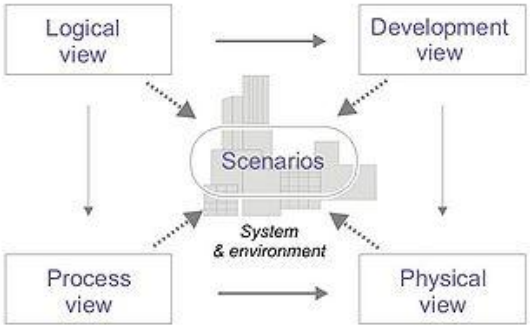
http://unity3d.com/support/documentation/
MSDN Visual Studios Documentation
http://msdn.microsoft.com/en-us/library/52f3sw5c%28v=VS.90%29.aspx

GLOSSARY

Term	Definition
4+1 Architectural Model	4+1 is a view model designed by Philippe Kruchten for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views.
UML	Unified Modeling Language is a standardized general-purpose modeling language in the field of software engineering. UML includes a set of graphical notation techniques to create abstract models of specific systems.
Unity	Unity is an integrated authoring tool for creating 3D video games or other interactive content such as architectural visualizations or real-time 3D animations.
OpenCV	OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real time computer vision, developed by Intel and now supported by Willow Garage.

ARCHITECTURAL REPRESENTATION

This document use Kruchten's 4+1 view model. This model breaks down the system into a set of views, each capturing a specific aspect of the system, address concerns for end users, developers, system integrators and system engineers:



View	Description
Use case view	Describes the end-user view of the system functionalities.
Logical view	An abstract description of the system's parts, contexts, and how they interact with each other to accomplish system functionalities.
Development view	Focuses on the actual software module organization, and implementation. It also addresses architecture goals and constraints and their solutions.
Process view	The process architecture takes into account non-functional requirements,

	such as scalability and availability. It addresses issues of concurrency and distribution.
Physical view	Describes deployment structures, configurations and communication protocols.

USE CASE VIEW

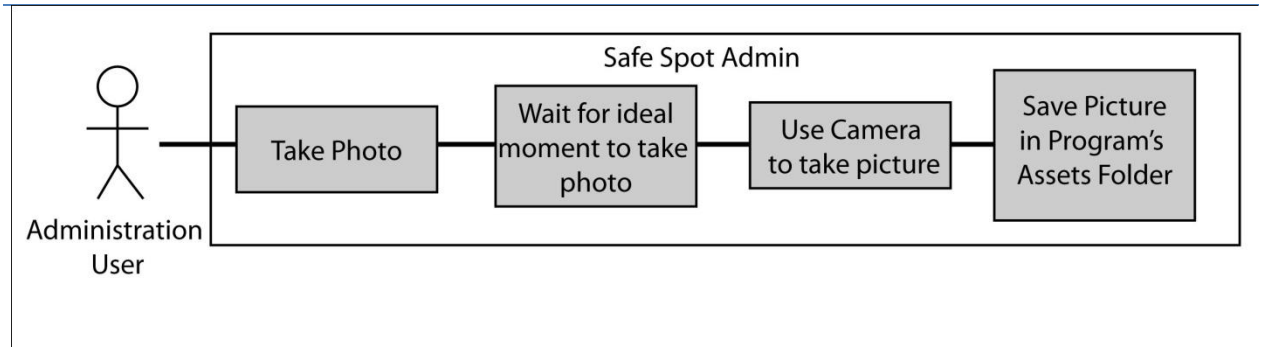
Use-Case diagrams show each of the Actor(s) goals or tasks in using the application. An Actor, as defined by the Rational Unified Process, is any person or system that is external to the system being implemented that the system interacts with. The Use Cases do not define implementation but rather outline the tasks the Actors want to accomplish.

ACTORS

Name	Description
Admin User	An admin user is one of a group of individuals who will use the Unity scene to create a photo of an unpopulated area.
Visitor User	The visitor user will be one of the individuals who partakes in the experience.
EDG System	The EDG System will communicate with the Unity Scene to create an interactive augmented reality experience.

USER USE CASES

ADMIN USER USE CASES – DIAGRAM

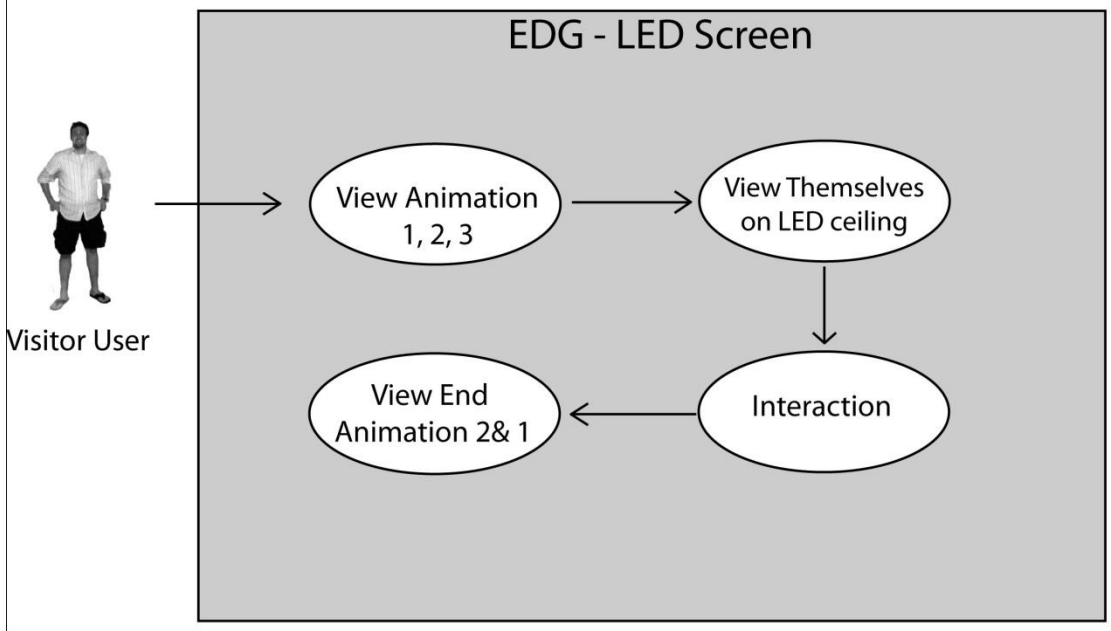


ADMIN USER USE CASES - DESCRIPTIONS

Name	Description
Take Photo	Take a photograph using the networked cameras.
Wait for idea moment to take photo	User will wait for the camera's target area to be unpopulated.
Use Camera to take picture	The user will take a photograph of the camera's target area. This photograph will be an individual frame from the stream, and should be obtainable via any third party program

	capable of capturing frames from an MJPEG stream.
Save Picture in Program's Assets Folder	The user will save the photograph in the program's assets folder.

VISITOR USER USE CASES - DIAGRAM



Event	Description
Animation 1	User sees a transparent ice animation overlaying the displayed background on the LED ceiling
Animation 2	User sees an opaque snow animation overlaying the displayed background on the LED ceiling
Animation 3	User sees an animation reveiling polar bears, iceberg shapes, and water areas displayed on the LED ceiling

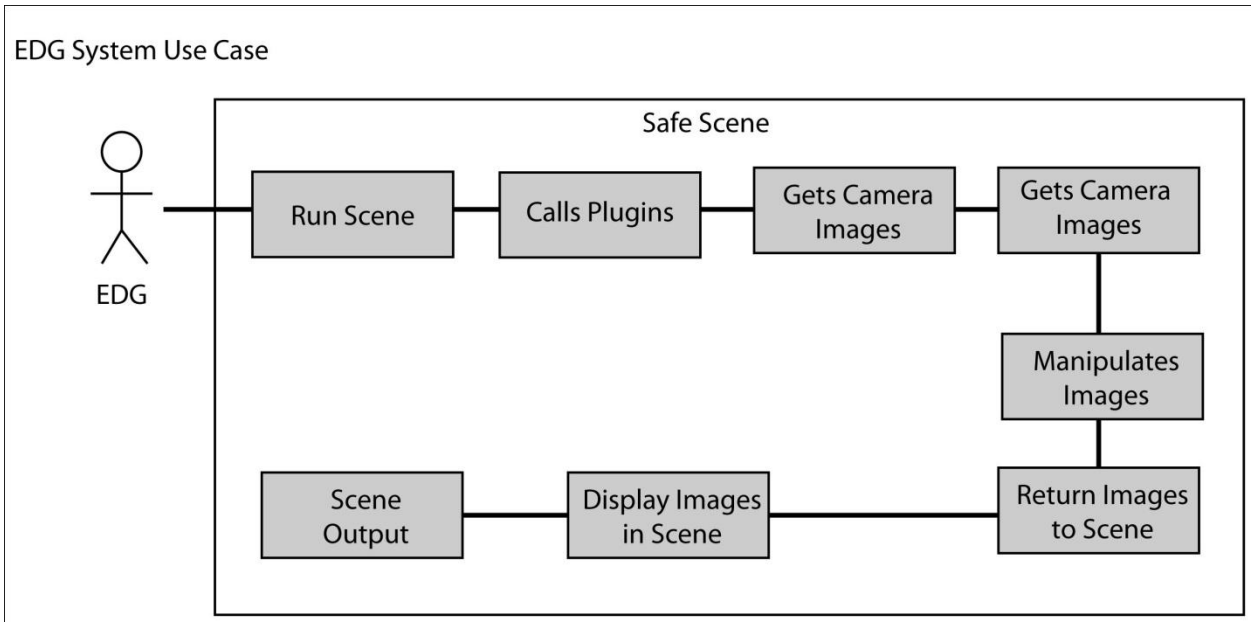
VISITOR USER USE CASES - DESCRIPTION

Name	Description
View Animation 1, 2, 3	Visitor views one animated feed at a time on the 3D display.
View Themselves on LED ceiling	Each visitor will see himself or herself on the LED display when she or he looks up at the ceiling.

Interaction	Each visitor will be able to interact with the experience that is displayed on the ceiling.
View End Animation 2 & 1	Each visitor will be able to view the snow and ice animations as the experience comes to an end.

SYSTEM USE CASES

EDG SYSTEM USE CASES – DIAGRAM



EDG SYSTEM USE CASES - DESCRIPTIONS

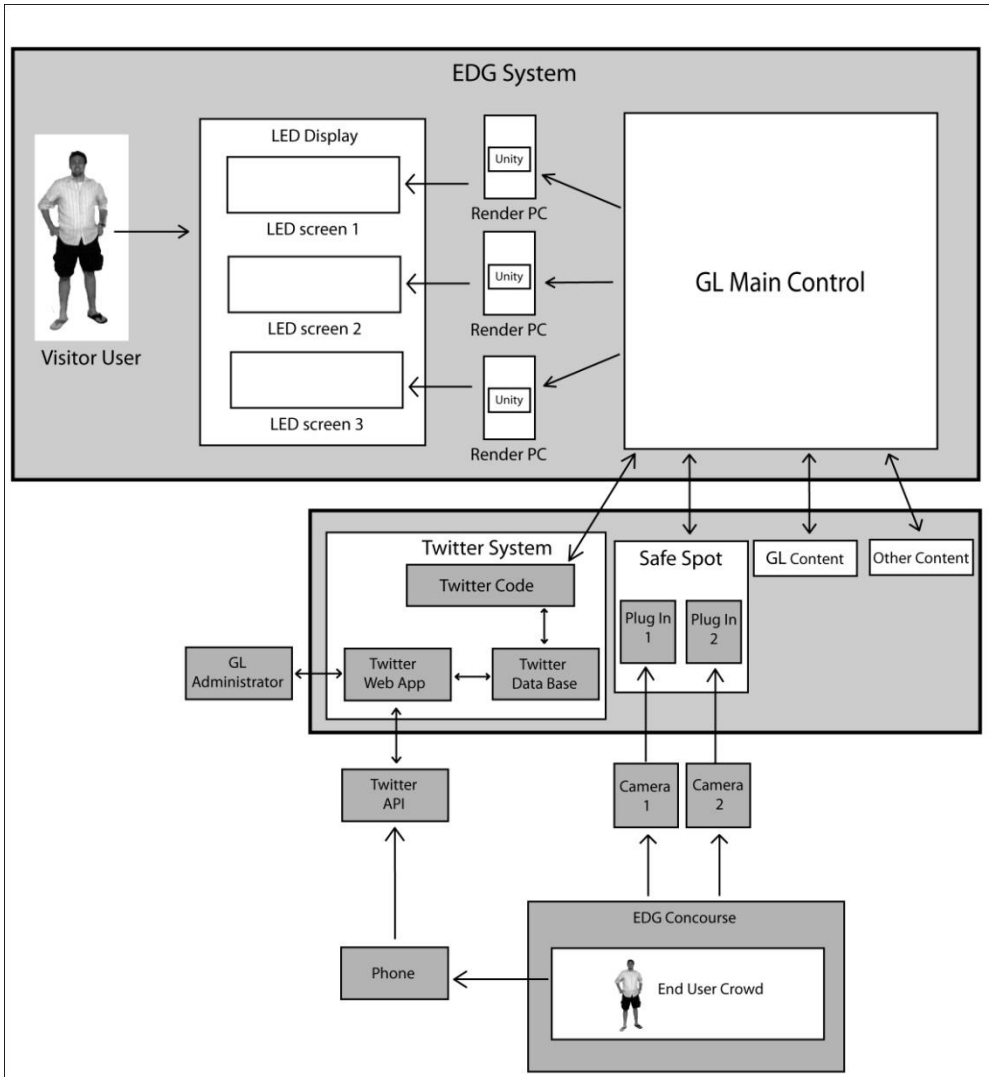
Name	Description
Run scene	The EDG System will run the Unity scene and then process the results for display.
Calls Plugins	The Unity scene will call two instances of the OpenCV plugin.
Gets Camera Images	One of the plugins will capture an image from its counterpart camera.
Gets Camera Images	One of the plugins will capture an image from its counterpart camera.
Manipulates Images	Each plugin will manipulate the respective image returned.

Return Images to Scene	The Unity scene will assemble the images and create the final display.
Scene Output	The EDG System will render the Unity scene’s output onto the ceiling display.

LOGICAL VIEW

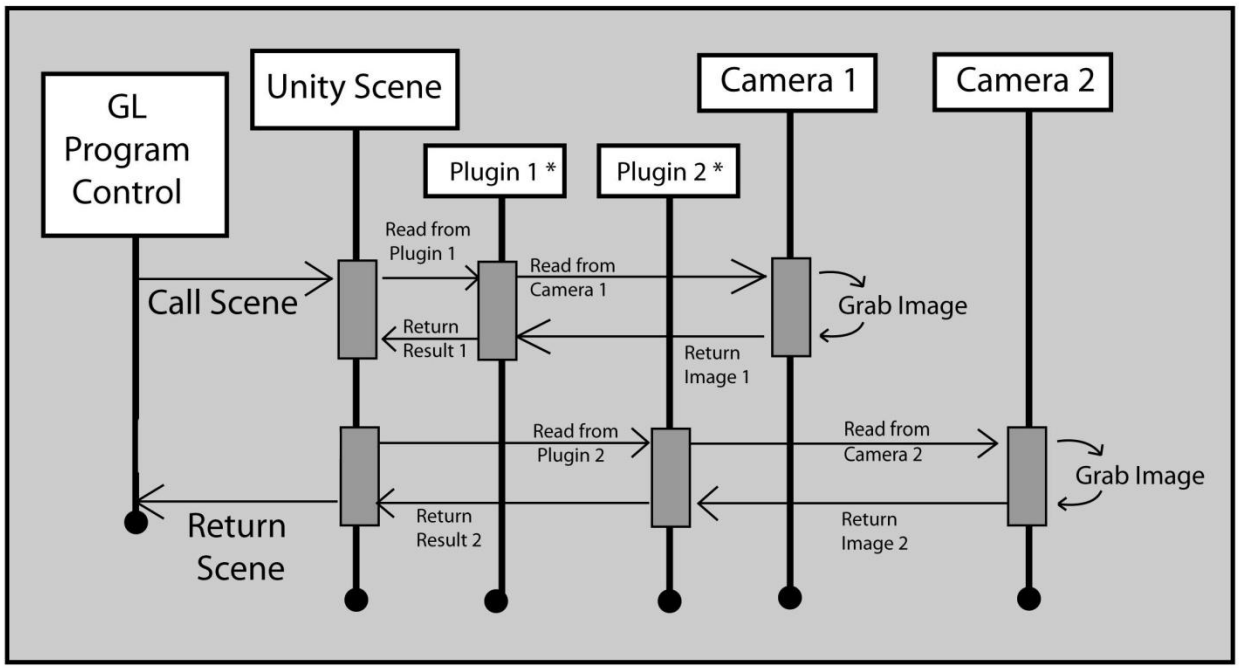
The following view is the Logical View of the solution being implemented.

LOGICAL SYSTEM DIAGRAM



SEQUENCE DIAGRAM FOR RUNNING UNITY SCENE - SAFE SPOT

Safe Spot
Logical View - Sequence Diagram



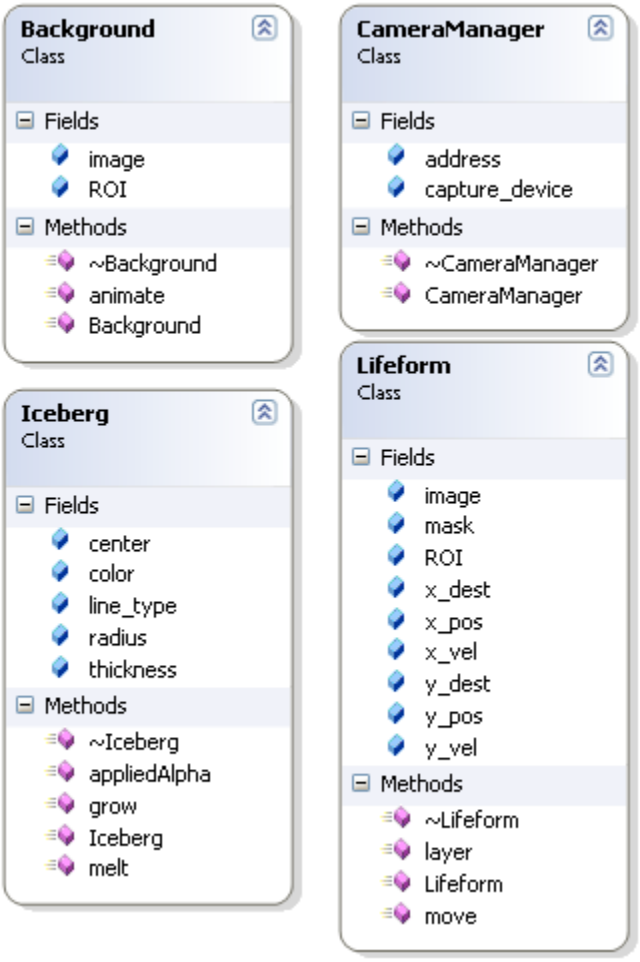
* Open CV Plugin

The Sequence Diagram attempts to explain the order in which processes operate and communicate with each other.

First, GL's program controller will invoke the Unity Scene, prompting for a display to render. In turn, the Unity Scene will call OpenCV Plugin 1 to request the first half of the display. Plugin 1 will read from Camera 1 in order to obtain an image from the camera so that it can manipulate the image to create an augmented reality frame. Result 1 (the manipulated image) will be returned to the Unity Scene. Next, the Unity Scene will call Plugin 2 in order to retrieve the second half of the display. Plugin 2 will grab an image from Camera 2 and then perform manipulations to create result 2. Result 2 will be returned to the Unity Scene so that it may be adjusted (probably rotated so that it matches the orientation of the first half of the display) and merged with the first half of the display to create the final display of the scene. GL's program controller will then handle the rendering of the scene's display on the LED screen.

CLASS DIAGRAM

OPENCV PLUGIN



UNITY SCENE

The Unity scene will make use of a script to communicate with the OpenCV plugin.

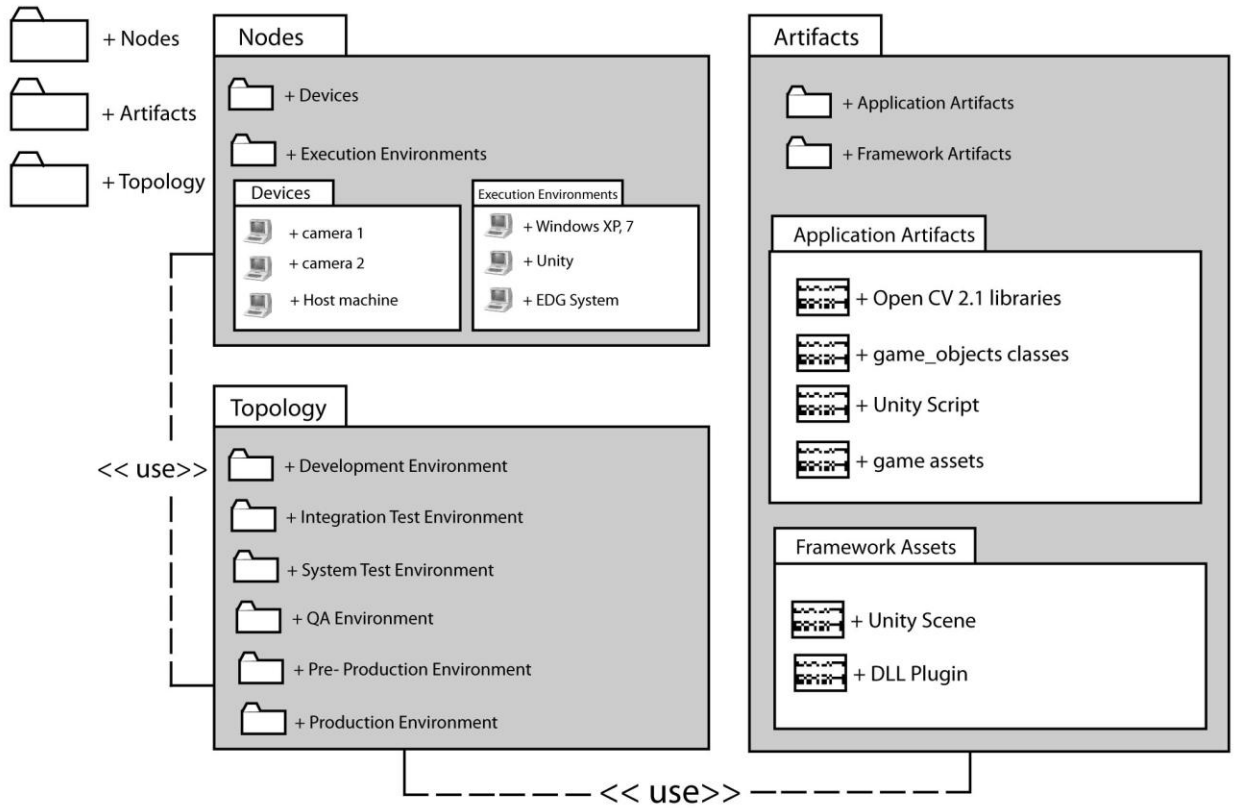
DEVELOPMENT VIEW

TECHNICAL STACK

DEVELOPMENT ENVIRONMENT

Windows XP Windows 7
Visual Studios 2008
Unity 3

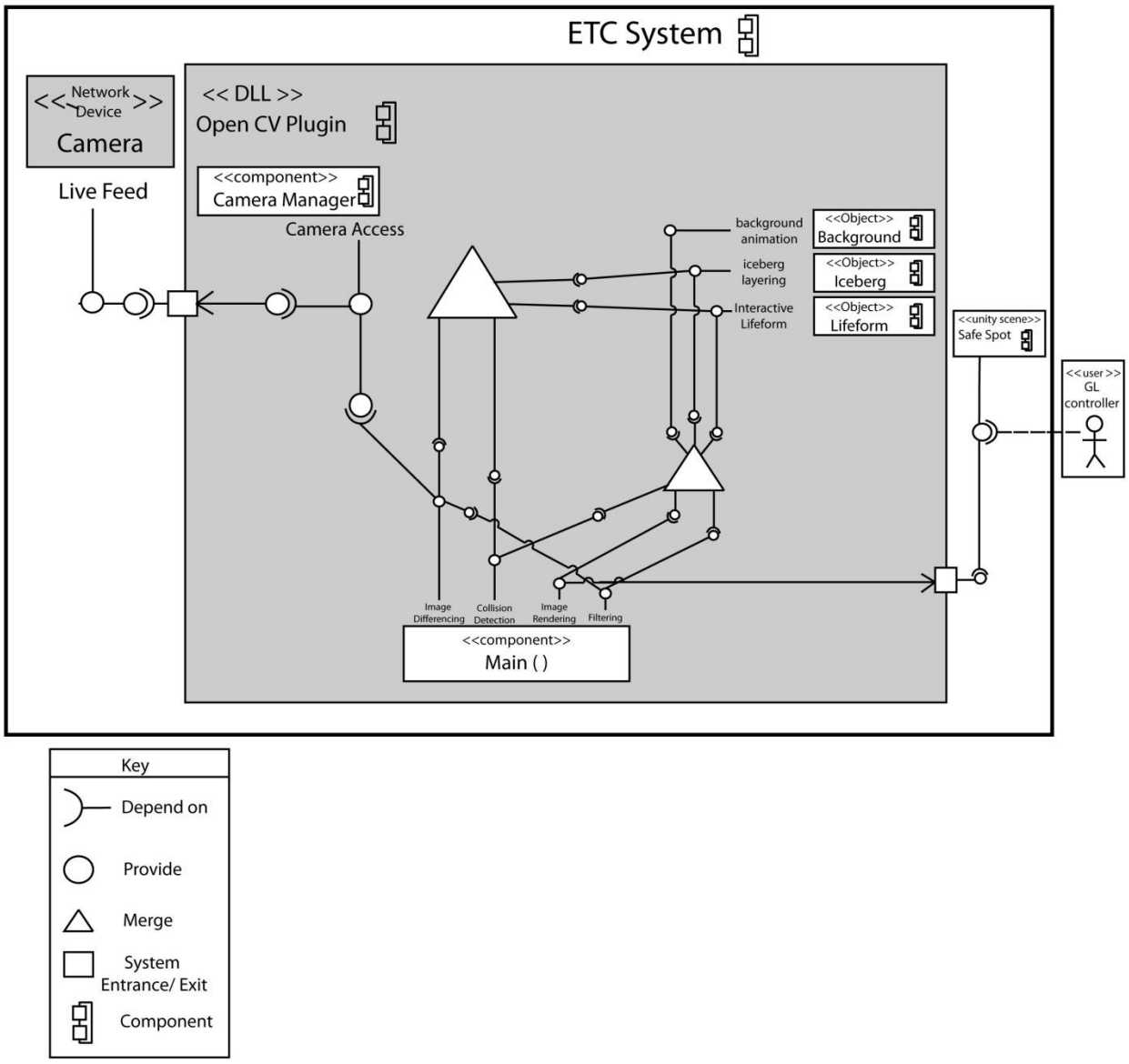
PACKAGE DIAGRAM



The Package Diagram attempts to depict the packages involved with the program.

COMPONENT DIAGRAM FOR THE SAFE SPOT SYSTEM

Safe Spot
Development View- Component Diagram

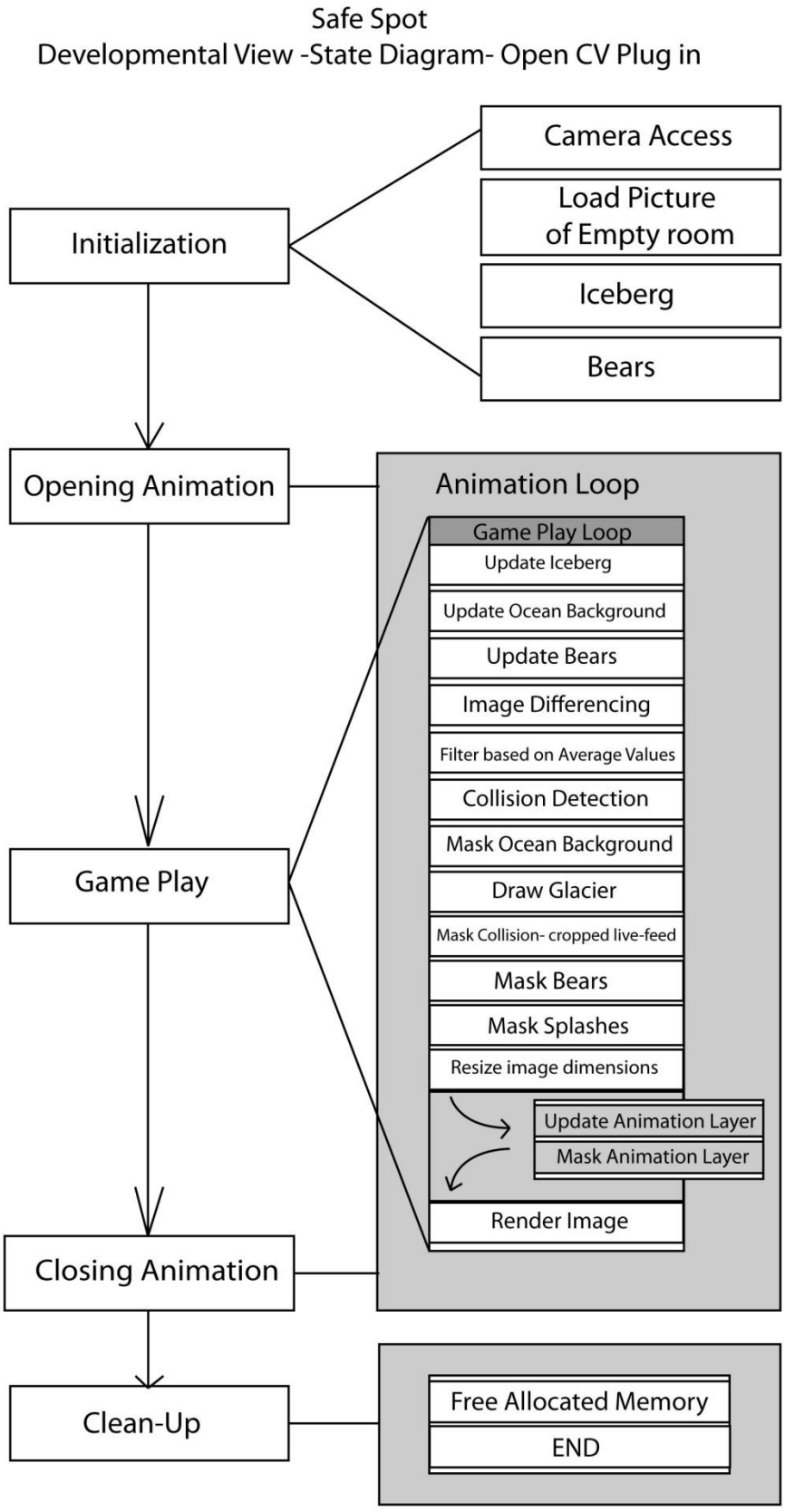


The Component Diagram attempts to depict how components are wired together to form larger components and subsystems.

GL's controller will make requests for the scene's display via functionality provided by the Unity Scene, Safe Spot. Safe Spot will be using the functionality of the Image Rendering provided by the OpenCV DLL Plugin to create a display. Image rendering requires the following dependencies: Collision Detection, Filtering, Background Animation, Iceberg Layering, and Interactive Lifeform instances. The Background

Animation, Iceberg Layering, and Interactive Lifeform instances are individual object assets with self-contained dependencies. Collision Detection will rely upon the functionality of Image Differencing, Iceberg Layering, and Interactive Lifeform instances. Image Differencing depends on functionality provided by the Camera Access of the Camera Manager component. The Camera Access will pull from the Live Feed of a network Camera external to the plugin. Filtering also depends on Camera Access.

STATE DIAGRAM FOR THE OPENCV PLUGIN



The State Diagram attempts to depict all of the states that the OpenCV Plugin will transition through during the lifetime of the plugin.

Initialization: The initialization phase will initialize the camera manager to access the network cameras, load the unpopulated ‘empty room’ photograph into memory, initialize the vector-based values of the Iceberg(s), and the initial animation and movement variables of the Bears.

Opening Animation: The Opening Animation phase will consist of the Game Play Loop (see Game Play) as well as a call to update the animation layer and a call to mask the layer on top of the other layers just before rendering the final image.

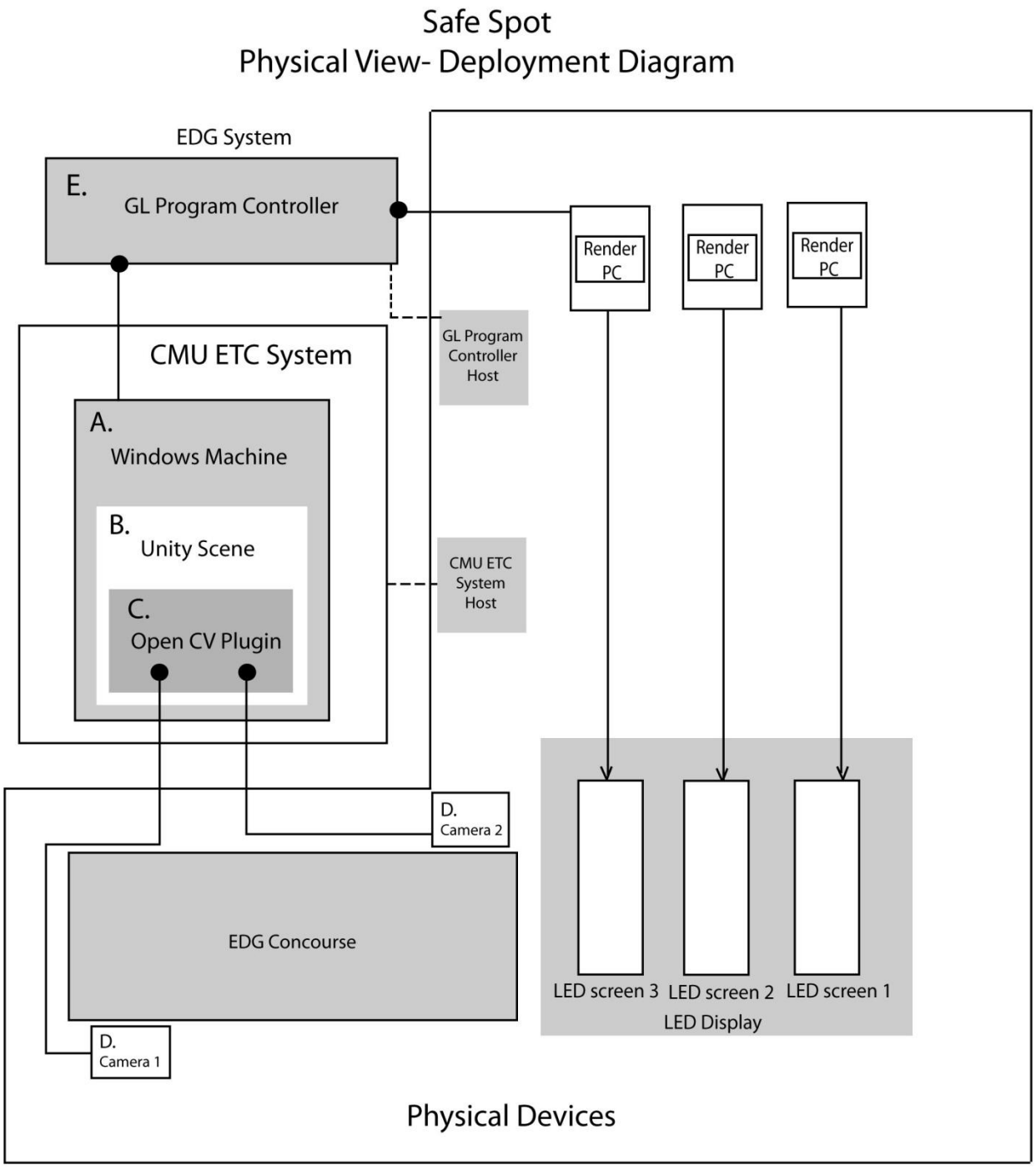
Game Play: The Game Play phase consists of the Game Play Loop, which comprises of updating game objects, image differencing ‘live’ images against the ‘empty room’ image, filtering against average values, performing collision checks with the interactive game objects, masking and layering, resizing the final image, and then rendering the image for output.

Closing Animation: The Closing Animation phase will behave just as the Opening Animation did with the exception that the animations used may be slightly different.

Clean-Up: The Clean-Up phase will result in the freeing of allocated memory and the end of the plugin’s execution.

PHYSICAL VIEW

DEPLOYMENT DIAGRAM FOR SAFE SPOT



The Deployment Diagram attempts to depict the physical and virtual components of the EDG System.

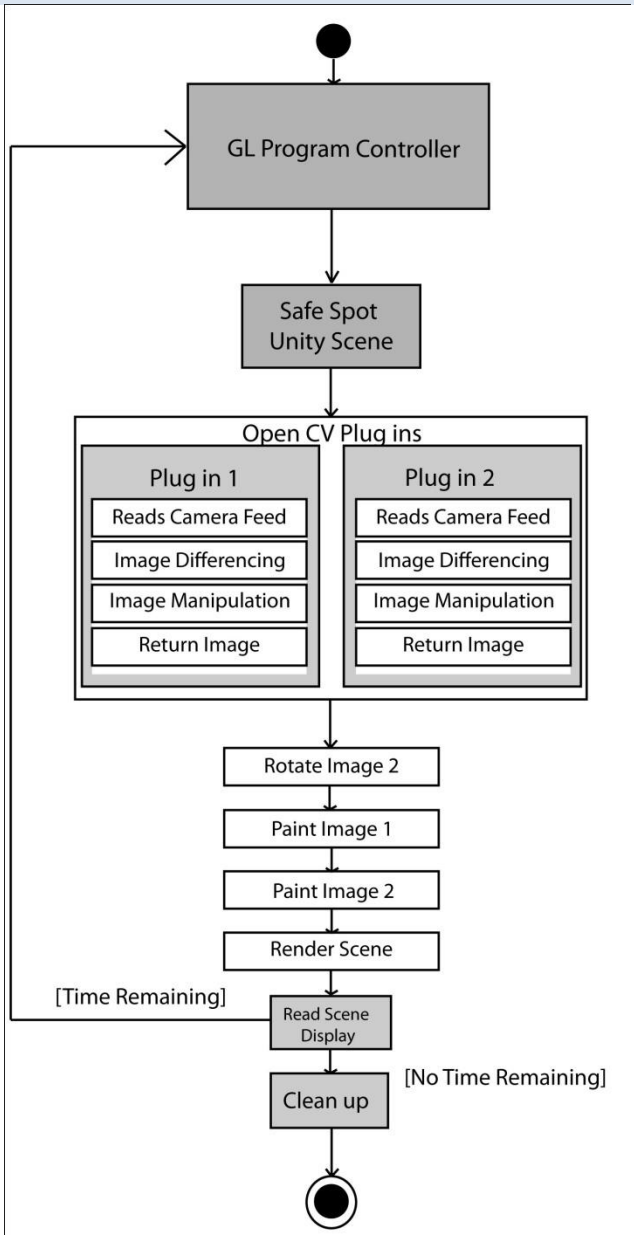
A. The CMU ETC System Host will have either the Windows XP or Windows 7 operating systems.
B. The Unity Scene (Safe Spot) will be running from within Unity or a Unity-like engine.
C. The OpenCV plugin will be called from within the Unity Scene
D. The cameras will be setup on a Local Area Network with a login address such as: http://username:password@address/cgi-bin/video.cgi?msubmenu=mjpg
The physical location of the cameras will be determined before deployment.

GL's role in the EDG System is not defined in this document, but the following are assumed to be true:

E. GL's Program Controller will mediate between the Safe Spot Unity Scene and EDG's render PCs.
F. The render PCs will handle rendering individual sections of the LED Display.
G. The LED Display will output the results of the render onto the ceiling.

PROCESS VIEW

ACTIVITY DIAGRAM



CONCURRENCY

Two instances of the OpenCV plugin will read from two cameras.

DISTRIBUTION

The deliverable will be a single Unity Scene which contains the OpenCV plugin.

INTEGRATORS

There may be a need for the integration of third party software so that the plugin can read from the cameras. Currently the prototype is using VLC media player to read from the cameras' network address, a VCam plugin for the media player, and a virtual camera emulator called VCam to produce a local output for the OpenCV plugin to read.

PERFORMANCE VARIABLES

Variables which can affect performance include the available memory and processors for the Unity scene to use, and the inclusion of a third party-based camera pipeline.

SCALABILITY

Crowd size will not affect the scalability of this program.

OTHER VARIABLES

Other variables which could affect the program include lighting, and the random numbers generated within the OpenCV plugin.

SECURITY

Camera access is currently expected to occur from an http request via a login name and password.